Claims listing:

1. (cancelled)

2. (previously presented) The method of claim ~~1~~ 50, wherein the at least one application state comprises a representation of a runtime snapshot of the application while under test.

3. (previously presented) The method of claim 2, wherein the at least one application state comprises a set of application objects, attributes of the application objects, and values of the attributes.

4. (previously presented) The method of claim 2, wherein the at least one application state is a plurality of application states, and the plurality of application states are arranged in a hierarchical manner.

5. (original) The method of claim 2, wherein the database system is a relational database management system.

6. (previously presented) The method of claim 2, wherein the database system is an extensible markup language (XML) database management system.

7. (previously presented) The method of claim 2, wherein the at least one scripting language is at least one of a typed programming language or an untyped programming language used for at least one of recording or authoring test cases.

8. (previously presented) The method of claim 2, wherein the at least one external interaction sequences comprises a representation of events invoked by external agents on the set of application objects.

9. (previously presented) The method of claim 8, wherein at least one external agent is at least one of a human agent or a software agent.

10. (previously presented) The method of claim 8, wherein the at least one interaction sequence comprises at least one flow control structure for capturing at least one of a sequential interaction, a concurrent interaction, a looping interaction, or a conditional interaction.

11. (previously presented) The method of claim 2, further comprising:

implementing at least one syntax analyzer for the at least one test case.

12. (previously presented) The method of claim 11, wherein a distinct syntax analyzer is implemented for each scripting language used in each test case.

13. (previously presented) The method of claim 12, wherein the at least one syntax analyzer comprises rules of syntax analysis that are provided in Extended Backus-Naur Form.

14. (previously presented) The method of claim 12, further comprising generating at least one parse tree in the form of an Abstract Syntax Tree using the at least one syntax analyzer.

15. (previously presented) The method of claim 2, wherein the semantic analysis is used to convert an Abstract Syntax Tree to the abstract representation based on an Application Object Model.

16. (previously presented) The method of claim 15, wherein the semantic analysis is used to decompose the Abstract Syntax Tree into the at least one application state, the at least one external interaction sequences and the input data.

17. (previously presented) The method of claim 15, wherein the Application Object Model comprises a metadata representation of the application.

18. (previously presented) The method of claim 17, wherein the metadata representation comprises object type definitions for application objects.

19. (previously presented) The method of claim 17, wherein the metadata representation comprises attribute definitions for each type of application object.

20. (previously presented) The method of claim 17, wherein the metadata representation comprises definitions of a plurality of methods and events that are supported by each type of application object.

21. (previously presented) The method of claim 17, wherein the metadata representation comprises definitions of a plurality of effects of events on an application state.

22. (previously presented) The method of claim 18, wherein the object type definitions comprise hierarchical object types, container object types, and simple object types.

23. (previously presented) The method of claim 22, wherein each hierarchical object type is associated with a distinct application state; and wherein each container object type comprises an application object type which is configured to contain instances of other application objects.

24. (previously presented) The method of claim 23, wherein the state associated with a hierarchical application object type is at least one of a modal application state or a nonmodal application state.

25. (previously presented) The method of claim 24, wherein the modal application state is configured to restrict possible interactions with application object instances available within a current application state.

26. (previously presented) The method of claim 22, wherein the effects of events on the at least one application state capture at least one consequence of the events to the application state.

27. (previously presented) The method of claim 26, wherein the at least one consequence of an event comprises at least one of creation of a new object instance of a given type, deletion of an object instance of a given type, modification of attributes of an existing object instance, or selection of an instance of an object type.

28. (previously presented) The method of claim 27, wherein the creation of a new object instance for a hierarchical object comprises creation of a new application state.

29. (previously presented) The method of claim 27, wherein the selection of an instance of an object type that is hierarchical comprises selection of the application state associated with the instance.

30. (previously presented) The method of claim 2, further comprising:

enriching the abstract representation based on information from an application metadata repository.

31. (previously presented) The method of claim 30, wherein enriching the abstract representation comprises extracting values for attributes of application objects associated with the at least one test case that are missing.

32. (previously presented) The method of claim 30, wherein enriching the abstract representation comprises decoupling the at least one test case from at least one of an associated recording environment or an associated authoring environment.

33. (previously presented) The method of claim 30, wherein enriching the abstract representation provides attributes that are stable within an application metadata representation.

34. (previously presented) The method of claim 33, wherein at least one application object is identified by an identification field within the application metadata repository.

35. (previously presented) The method of claim 33 wherein the identification field provides platform independence of the abstract representation.

36. (previously presented) The method of claim 35, wherein enriching the abstract representation provides a representation of test cases that is independent of any particular test execution environment.

37. (previously presented) The method of claim 2, further comprising:

separating application object attributes and input data from external interaction sequencing to provide automatic parameterization.

38. (cancelled)

39. (previously presented) The system of claim 51, wherein the at least one test case is converted using a syntax analyzer.

40. (previously presented) The system of claim 51, wherein the semantic analysis is configured to convert an abstract syntax tree to the abstract representation based on an Application Object Model.

41. (canceled)

42. (previously presented) The system of claim 51, further comprising:

a third set of instructions which, when executed by the processor, configures the processor to enrich the abstraction representation to provide a representation of the at least one test case that is independent of any particular test execution environment.

43. (cancelled)

44. (previously presented) The system of claim 52, wherein the first set of instructions comprises a syntax analyzer .

45. (previously presented) The system of claim 44, wherein the syntax analyzer is configured to generate a parse tree in the form of an Abstract Syntax Tree based on the at least one test case.

46. (previously presented) The system of claim 45, wherein the first set of instructions, when executed by the processor, further configures the processor to convert the Abstract Syntax Tree to the abstract representation based on an Application Object Model.

47. (previously presented) The system of claim 52, wherein the first set of instructions, when executed by the processor, further configures the processor to enrich the abstract representation using information from an application metadata repository.

48. (previously presented) The system of claim 52, wherein the first set of instructions, when executed by the processor, further configures the processor to separate the

attributes and the input data from the at least one external interaction sequence to provide automatic parameterization.

49. (cancelled).

50. (currently amended) A method for transforming test cases that are converted from a source test script to an abstract representation and storing abstract representation of test cases into a [database system] data store, comprising:

storing test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases;

importing at least one test case[[s]] written in one or more scripting language[[s]];

~~using semantic analysis to convert test cases to an abstract representation that includes application state, external interaction sequences and input data without changing or deleting an original test case based on an application object model~~ converting the at least one test case to an abstract representation using semantic analysis without changing or deleting an original test case based on an application object model, the abstract representation having at least three separate components including an application state, external interaction sequences and input data, where the application object mode is a metadata representation for a modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state[[.]] and

storing the abstract representation into a database; and

generating at least one platform-specific test script based on the abstract representation stored in the database using a platform-specific mapping, the platform-specific mapping including a language mapping and an environmental mapping.

51. (currently amended) A system for transforming test cases that are converted from a source test script to an abstract representation and storing abstract representation of test cases into a data store, comprising:

a processor for importing test cases written in one or more scripting languages;

a database for storing abstract representations;

logic that ~~uses semantic analysis to convert test cases to an abstract representation that includes application state, external interaction sequences and input data without changing or deleting an original test case based on an application object mode~~ converts the at least one test case to an abstract representation using semantic analysis without changing or deleting an original test case based on an application object model, the abstract representation including an application state, external interaction sequences and input data, where the application object model is a metadata representation for modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state, ~~the logic using environment mappings providing platform independence of test cases and test scripts are generated for multiple test execution environments without changing or deleting an original test case, the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts~~ the logic generating at least one platform-specific test script based on the abstract representation stored in the database using a platform-specific mapping, the platform-specific mapping including a language mapping and an environmental mapping.

52. (currently amended) A computer system for transforming test cases that are converted from a source test script to an abstract representation and storing abstract representation of test cases into a data store, comprising:

a processor; [[and]]

a database for storing abstract representations;

a memory coupled to the processor, the memory storing ~~test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases, importing test cases written in one or more scripting languages and uses semantic analysis to convert test cases to an abstract representation that includes application state, external interaction sequences and input data without changing or deleting an original test case based on an application object mode, where the application object model is a metadata representation for modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state~~ the at least one test case that is converted to an abstract representation through semantic analysis without changing or deleting an original test case based on an application object model, the abstract representation including an application state, external interaction sequences and input data, where the application object model is a metadata representation for modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state, the memory storing at least one platform-specific test script generated based on the abstract representation stored in the database by use of a platform-specific mapping, the platform-specific mapping including a language mapping and an environmental mapping.